Выпускная квалификационная работа

по курсу

«Data Science»

по теме:

Прогнозирование конечных свойств композитных материалов

Слушатель: Каминский Александр

Этапы работы

- Разведочный анализ данных
- Удаление выбросов
- Анализ признаков и визуализация с целью выявления зависимостей
- Предобработка данных
- Разработка и обучение регрессионных моделей для прогнозирования «Модуль упругости при растяжении, ГПА» и «Прочность при растяжении»
- Нейронная сеть для рекомендации «Соотношение матрица-наполнитель»
- Разработка приложения Flask

Разведочный анализ данных

Даны 2 файла: **X_bp.xlsx** (с данными о параметрах, состоящий из 1023 строк и 10 столбцов данных) и **X_nup.xlsx** (данными нашивок, состоящий из 1040 строк и 3 столбцов данных). После объединения - 1023 строки, 13 столбцов.

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	210.000000	70.000000	3000.000000	220.000000	0	4.000000	57.000000
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000	0	4.000000	60.000000
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	210.000000	70.000000	3000.000000	220.000000	0	4.000000	70.000000
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	210.000000	70.000000	3000.000000	220.000000	0	5.000000	47.000000
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000	0	5.000000	57.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	209.198700	73.090961	2387.292495	125.007669	90	9.076380	47.019770
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	350.660830	72.920827	2360.392784	117.730099	90	10.565614	53.750790
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	740.142791	74.734344	2662.906040	236.606764	90	4.161154	67.629684
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	641.468152	74.042708	2071.715856	197.126067	90	6.313201	58.261074
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	758.747882	74.309704	2856.328932	194.754342	90	6.078902	77.434468

1023 rows × 13 columns

Разведочный анализ данных

File Edit View Insert Cell Kernel Widgets Help Не доверять Python 3 (ipykernel)

Запуск Код

Ввод [5]:
```
# Получим описательную статистику и транспонируем для удобства доступа к названиям колонок.
a =df.describe()
a.T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Разведочный анализ данных

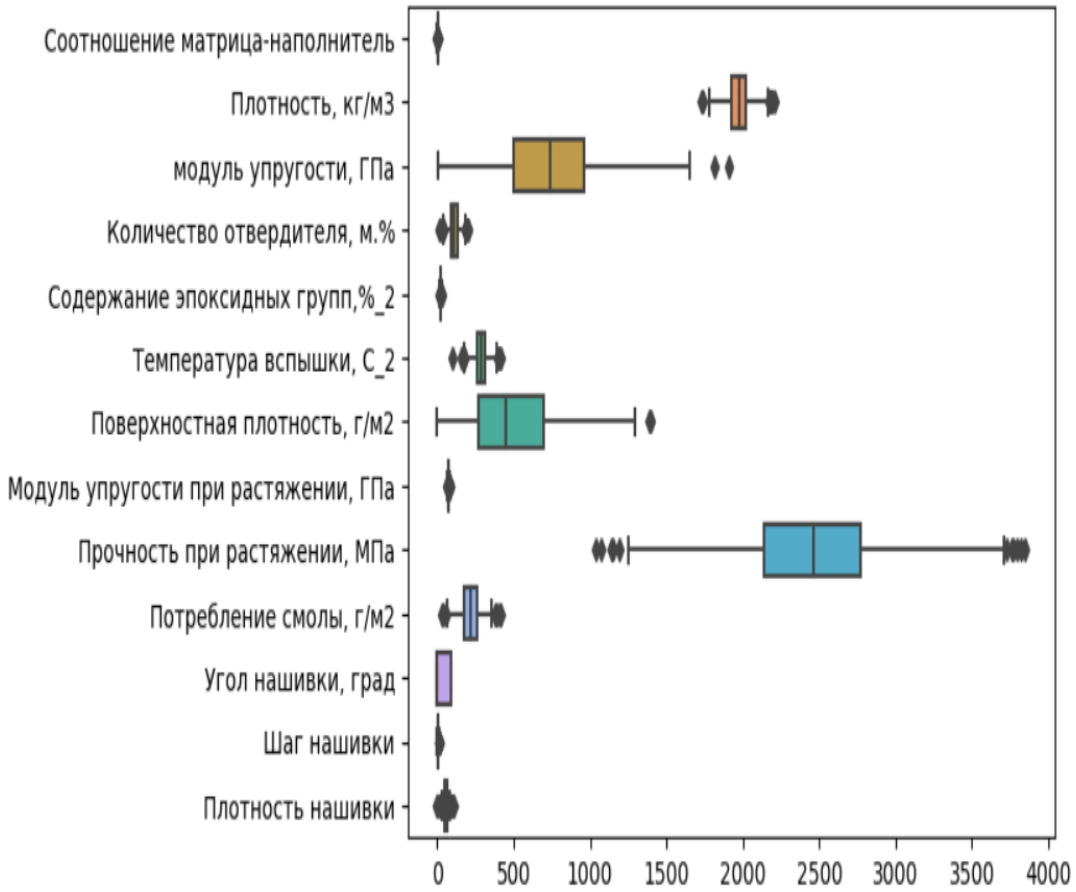
t View Insert Cell Kernel Widgets Help

Запуск Код

```
9]: # оценка наличия,количества выбросов в столбцах исходного dataset
    plt.figure(figsize=(20, 40))
    i=1
    for name in df.columns:
        plt.subplot(5,3,i)
        sns.boxplot(y=df[name], color = 'g')
        outlier = boxplot_stats(df[name])
        print ('Количество выбросов в столбце ', name, ': ', len(outlier[0]['fliers']))
        i +=1
```

```
Количество выбросов в столбце  Соотношение матрица-наполнитель :  6
Количество выбросов в столбце  Плотность, кг/м3 :  9
Количество выбросов в столбце  модуль упругости, ГПа :  2
Количество выбросов в столбце  Количество отвердителя, м.% :  14
Количество выбросов в столбце  Содержание эпоксидных групп,%_2 :  2
Количество выбросов в столбце  Температура вспышки, С_2 :  8
Количество выбросов в столбце  Поверхностная плотность, г/м2 :  2
Количество выбросов в столбце  Модуль упругости при растяжении, ГПа :  6
Количество выбросов в столбце  Прочность при растяжении, МПа :  11
Количество выбросов в столбце  Потребление смолы, г/м2 :  8
Количество выбросов в столбце  Угол нашивки, град :  0
Количество выбросов в столбце  Шаг нашивки :  4
Количество выбросов в столбце  Плотность нашивки :  21
```

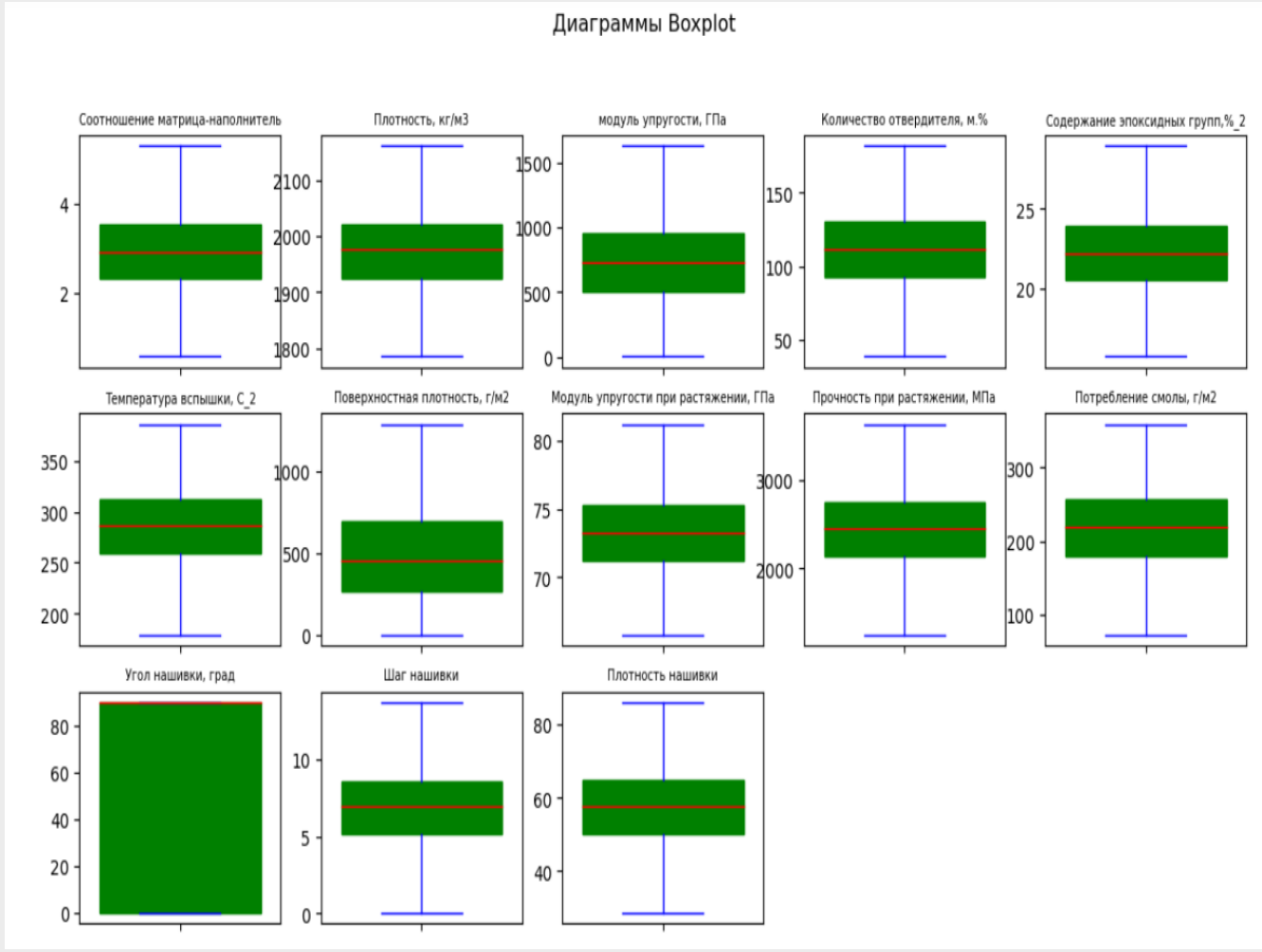
```
sns.boxplot(data = df, orient="h")
plt.show()
```



Удаление выбросов

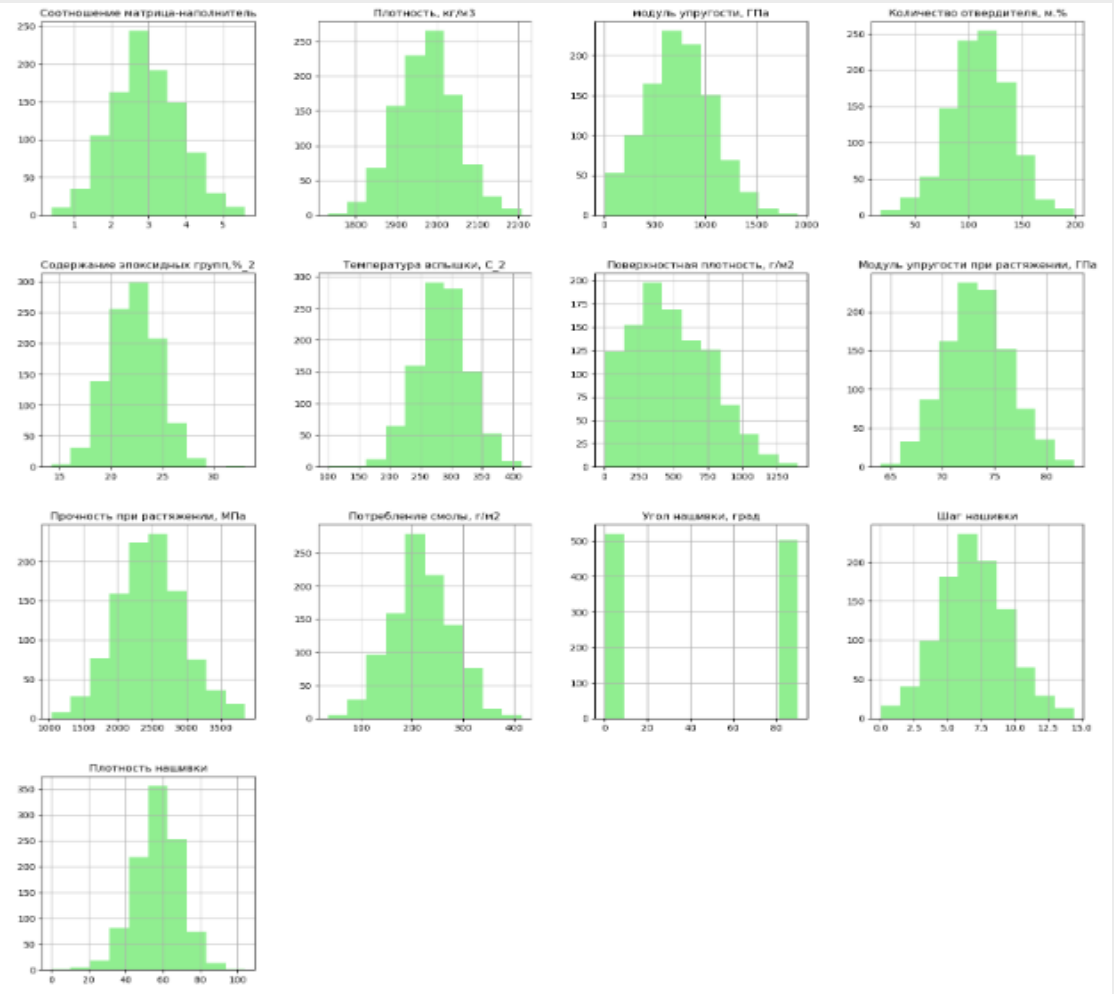
```
# оценка наличия, количества выбросов в очищенном 3 раза датасете
plt.figure(figsize=(20, 40))
i=1
for name in df_clean.columns:
    plt.subplot(5,3,i)
    sns.boxplot(y=df_clean[name], color = 'g')
    outlier = boxplot_stats(df_clean[name])
    print ('Количество выбросов в столбце ', name, ': ', len(outlier[0]['fliers']))
    i +=1
```

```
Количество выбросов в столбце  Соотношение матрица-наполнитель :  0
Количество выбросов в столбце  Плотность, кг/м3 :  0
Количество выбросов в столбце  модуль упругости, ГПа :  0
Количество выбросов в столбце  Количество отвердителя, м.% :  0
Количество выбросов в столбце  Содержание эпоксидных групп,%_2 :  0
Количество выбросов в столбце  Температура вспышки, С_2 :  0
Количество выбросов в столбце  Поверхностная плотность, г/м2 :  0
Количество выбросов в столбце  Модуль упругости при растяжении, ГПа :  0
Количество выбросов в столбце  Прочность при растяжении, МПа :  0
Количество выбросов в столбце  Потребление смолы, г/м2 :  0
Количество выбросов в столбце  Угол нашивки, град :  0
Количество выбросов в столбце  Шаг нашивки :  0
Количество выбросов в столбце  Плотность нашивки :  0
```

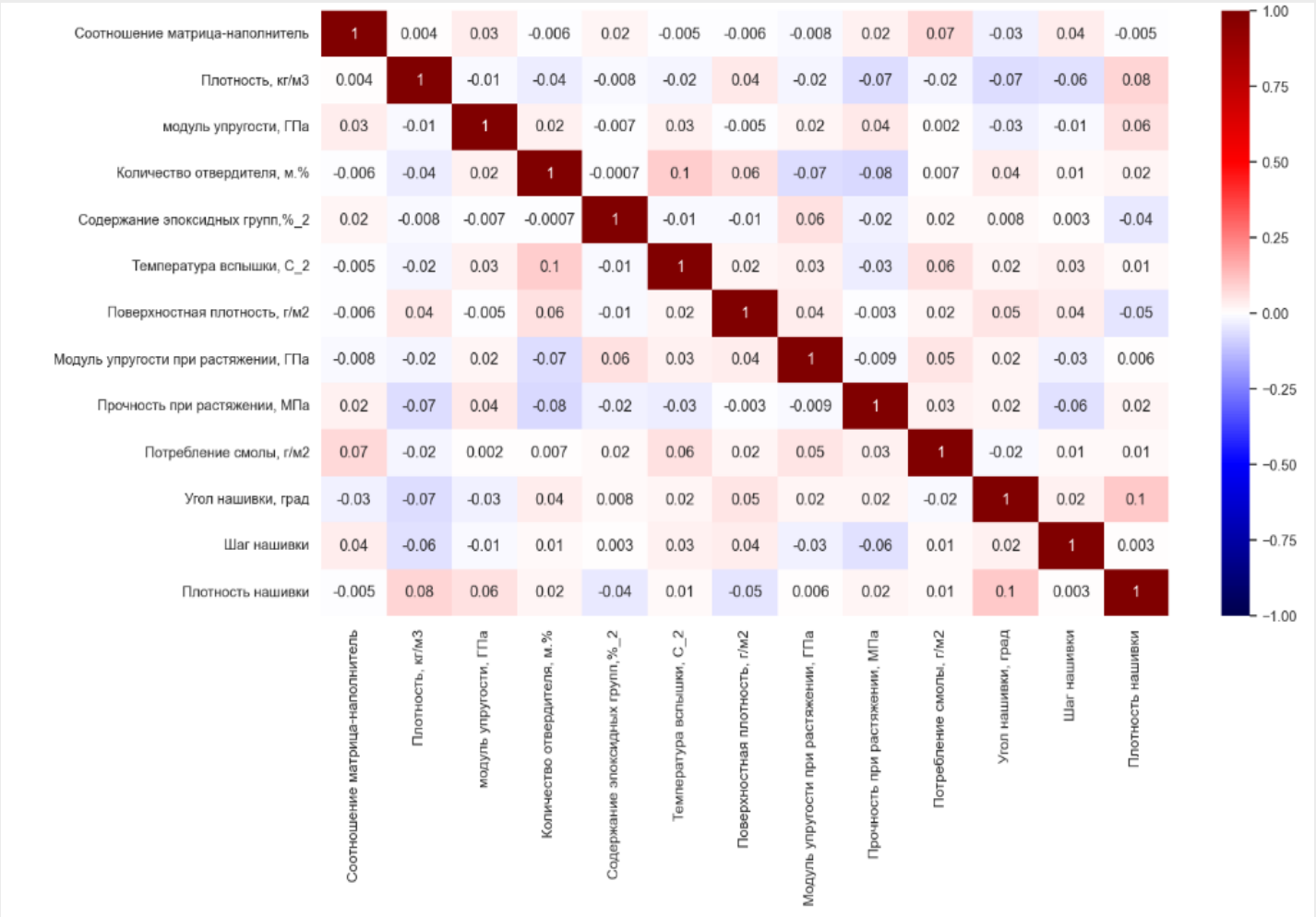


После удаления всех выбросов осталась 921 строка с данными

Анализ признаков и визуализация с целью выявления зависимостей

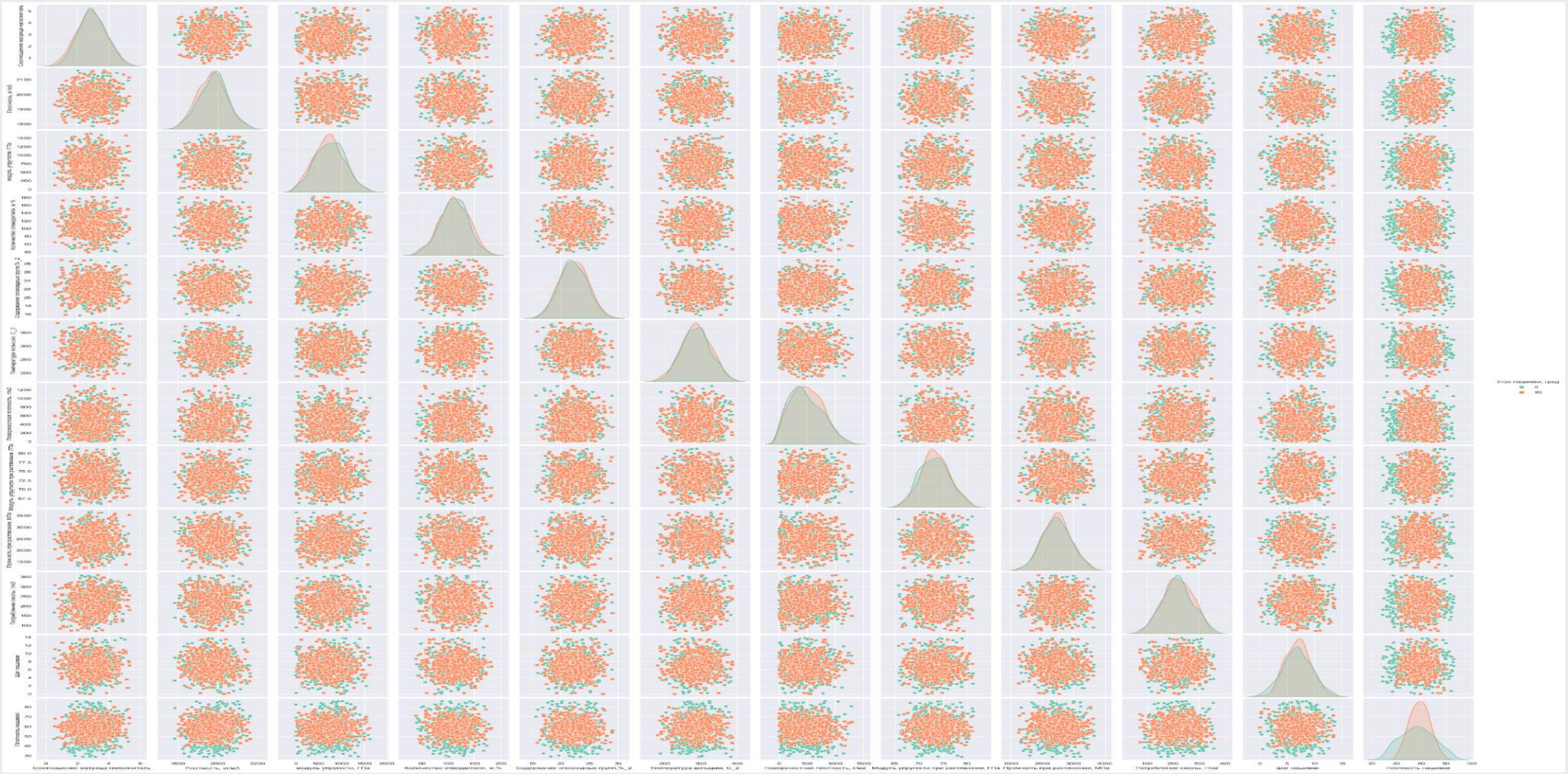


Гистограмма распределения



Тепловая карта

Анализ признаков и визуализация с целью выявления зависимостей



Попарные графики рассеяния точек с выделением значений Угол нашивки

Разработка и обучение регрессионных моделей

На примере Ridge регрессии: выбор модели, подбор гиперпараметров по сетке (GridSearchCV) с перекрестной проверкой (cross validation K-fold), обучение модели, денормализация предсказанных значений и оценка результатов при помощи метрик MAE и R2, запись результатов для итоговой таблицы

Ridge regression

Гиперпараметры для поиска по сетке

Ввод [15]:
```
params = [{'alpha': [20, 10, 1, 0.1, 0.01, 0.0001],
           "solver": ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']}]
```

Поиск по сетке с перекрестной проверкой

Ввод [16]:
```
modelRidge = Ridge()
cv_scheme = KFold(n_splits=10, shuffle=True, random_state=1)
cv = GridSearchCV(estimator=modelRidge, param_grid=params,
                  scoring='neg_root_mean_squared_error', cv=cv_scheme, return_train_score=True, n_jobs=-1)

cv.fit(X_train_norm,y_train_norm)
```

Out[16]:
```
GridSearchCV(cv=KFold(n_splits=10, random_state=1, shuffle=True),
             estimator=Ridge(), n_jobs=-1,
             param_grid=[{'alpha': [20, 10, 1, 0.1, 0.01, 0.0001],
                          'solver': ['svd', 'cholesky', 'lsqr', 'sparse_cg',
                                     'sag', 'saga']}],
             return_train_score=True, scoring='neg_root_mean_squared_error')
```

Ввод [17]:
```
print("Лучший параметр для гребневой регрессии:")
print(cv.best_params_)
print("Лучший средний балл перекрестной проверки:")
print(cv.best_score_)
```

Ввод [17]:
```
print("Лучший параметр для гребневой регрессии:")
print(cv.best_params_)
print("Лучший средний балл перекрестной проверки:")
print(cv.best_score_)
```

```
Лучший параметр для гребневой регрессии:
{'alpha': 20, 'solver': 'sag'}
Лучший средний балл перекрестной проверки:
-0.19174330048567476
```

Ввод [18]:
```
# модель линейной регрессии Ridge
alpha = cv.best_params_['alpha']
solver = cv.best_params_['solver']

modelRidge = Ridge(alpha=alpha, solver=solver)
modelRidge.fit(X_train_norm,y_train_norm)
print (modelRidge.predict(X_test_norm).shape)
y_pred =  scaler_norm_y.inverse_transform (modelRidge.predict(X_test_norm))
MAERidge_1 = mean_absolute_error(y_test.iloc[:,0],y_pred[:,0])
MAERidge_2 = mean_absolute_error(y_test.iloc[:,1],y_pred[:,1])
R2Ridge_1 = r2_score(y_test.iloc[:,0],y_pred[:,0])
R2Ridge_2 = r2_score(y_test.iloc[:,1],y_pred[:,1])
print (MAERidge_1)
print (MAERidge_2)
print (R2Ridge_1)
print (R2Ridge_2)
```

```
(277, 2)
383.54244899027884
2.455318376517251
0.002727562086469071
0.0012387346725416526
```

Ввод [19]:
```
# записываем данные об ошибках в итоговую таблицу
MAE_y1.append(['Ridge_norm',MAERidge_1])
MAE_y2.append(['Ridge_norm',MAERidge_2])
R2_y1.append(['Ridge_norm',R2Ridge_1])
R2_y2.append(['Ridge_norm',R2Ridge_2])
```

Ввод [20]:
```
print(modelRidge.coef_)
```

```
[[-0.00861863 -0.0279491   0.02678046 -0.04696982 -0.00216113 -0.01310681
  -0.00824067  0.00212136 -0.00524504 -0.01594748  0.01973426]
 [-0.0057319  -0.01273972  0.02296786 -0.02266175  0.01996864  0.02058247
   0.01206452  0.02869083  0.01760028 -0.02169592  0.00961241]]
```

Разработка и обучение регрессионных моделей

Оценка результатов работы моделей с использованием MAE и R2

Out[48]:

	Модель регрессии	MAE Модуль упругости при растяжении, ГПа	MAE Прочность при растяжении, МПа
0	Ridge_norm	383.50	2.455078
1	Lasso_norm	383.00	2.457031
2	Elastic_norm	383.50	2.451172
3	GBR_norm	393.50	2.531250
4	KNeighborsRegressor_norm	391.25	2.632812
5	DecisionTreeRegressor_norm	392.75	2.724609
6	RandomForestRegressor_norm	384.25	2.478516
7	AdaBoostRegressor_norm	399.00	2.603516
8	NeuralNetwork_norm	387.50	2.451172

Out[49]:

	Модель регрессии	R2 Модуль упругости при растяжении, ГПа	R2 Прочность при растяжении, МПа
0	Ridge_norm	0.002728	0.001239
1	Lasso_norm	0.002850	-0.001080
2	Elastic_norm	-0.000235	-0.000864
3	GBR_norm	-0.049255	-0.065063
4	KNeighborsRegressor_norm	-0.051453	-0.197266
5	DecisionTreeRegressor_norm	-0.090393	-0.278809
6	RandomForestRegressor_norm	0.001158	-0.017715
7	AdaBoostRegressor_norm	-0.062561	-0.145752

Нейронная сеть для рекомендации «Соотношение матрица-наполнитель»

Ввод [7]:
```
#разбиение данных на тестовую и тренировочную часть
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42, shuffle=True)
```

Ввод [8]:
```
# нормализация  данных
scaler_norm = MinMaxScaler()
scaler_norm.fit(X)
Xnorm = pd.DataFrame (data =scaler_norm.transform(X), columns=X.columns)
scaler_norm_y = MinMaxScaler()
scaler_norm_y.fit(y)
ynorm = pd.DataFrame (data = scaler_norm_y.transform(y), columns=y.columns)
```

Ввод [9]:
```
#разбиение данных на тестовую и тренировочную часть
X_train_norm, X_test_norm, y_train_norm, y_test_norm = train_test_split(Xnorm,ynorm, test_size=0.3, random_state=42, shuffle=True
```

Out[28]:

	Версия нейросети	MAE Соотношение матрица-наполнитель
0	Нейросеть 0	0.715820
1	Нейросеть 1	0.714355
2	Нейросеть 2	0.713379
3	Нейросеть 3	0.714355
4	Нейросеть 4	0.713379
5	Нейросеть 5	0.713379
6	Нейросеть 6	0.713867
7	Нейросеть 7	0.713379

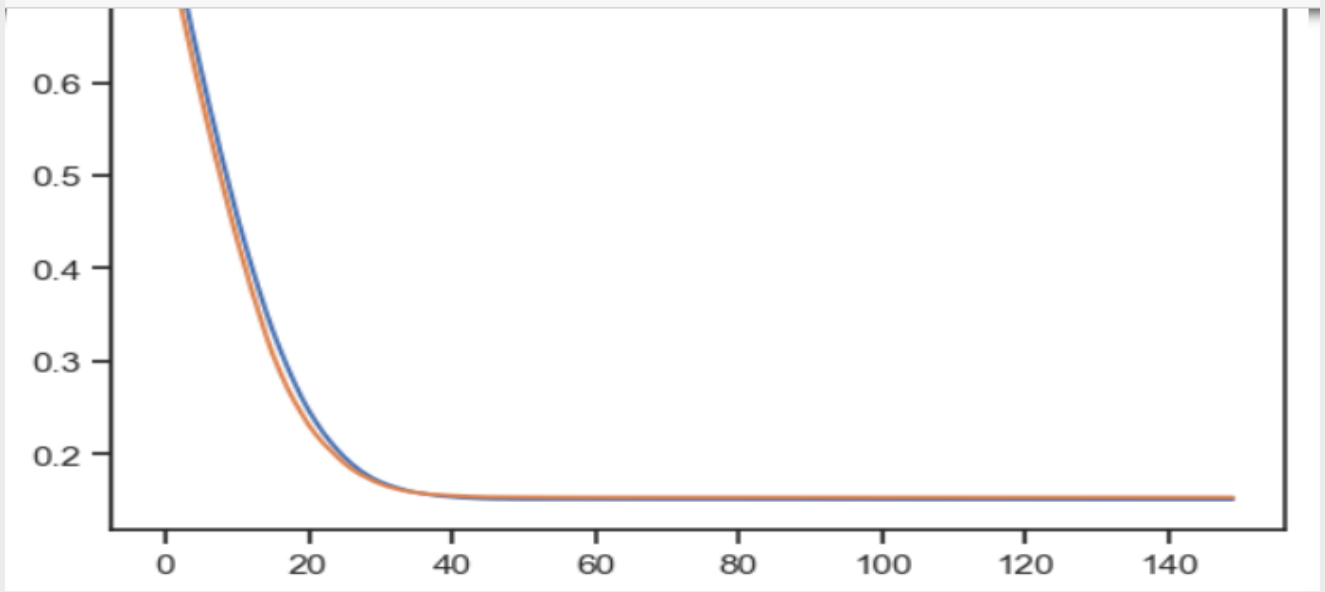
Нейронная сеть для рекомендации «Соотношение матрица-наполнитель»

Ввод [22]:

```
# 5
model_y5= Sequential()
model_y5.add(Dense(100, input_dim=12, activation='sigmoid'))
model_y5.add(LeakyReLU(alpha=1.0))
model_y5.add(Dense(50, activation='sigmoid'))
model_y5.add(LeakyReLU(alpha=1.0))
model_y5.add(Dense(1, activation='softmax'))

#IMPORTANT PART
model_y5.add(Dense(1, activation='linear'))
model_y5.compile(optimizer='adam', loss='mse', metrics=['mae'])

model_y5.summary()
history = model_y5.fit(X_train_norm,y_train_norm,
                       epochs=150,
                       validation_split=0.1,
                       verbose=2)
plt.plot(history.history['mae'], label = 'Точность train')
plt.plot(history.history['val_mae'], label = 'Точность test')
plt.xlabel = ('Epochs')
plt.ylabel = ('mae')
plt.legend()
plt.show()
```



Разработка приложения Flask

Ввод [*]:
```
from flask import Flask, request, render_template

import tensorflow as tf

app = Flask(__name__)

def prediction(params):
    model = tf.keras.models.load_model('models/mn_model_nn')
    pred = model.predict([params])
    return pred

@app.route('/', methods=['POST', 'GET'])
def predict():
    message = ''
    if request.method == 'POST':
        param_list = ('Плотность, кг/м3', 'модуль упругости, ГПа', 'Количество отвердителл, м.%',
                      'Содержание эпоксидных групп,%_2', 'Температура вспышки, С_2', 'Поверхностная плотность, г/м2',
                      'Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа', 'Потребление смолы, г/м2',
                      'Угол нашивки, град', 'Шаг нашивки', 'Плотность нашивки')
        params = []
        for i in param_list:
            param = request.form.get(i)
            params.append(param)
        params = [float(i.replace(',', '.')) for i in params]

        message = f'Соотношение матрица-наполнитель: {prediction(params)}'
    return render_template('mn.html', message=message)

if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Ввод [ ]:

Разработка приложения Flask

Расчет соотношения матрица-наполнитель

Введите параметры

2020	Плотность, кг/м3
735	Модуль упругости, ГПа
45	Количество отвердителя, м.%
22	Содержание эпоксидных групп,%_2
280	Температура вспышки, С_2
202	Поверхностная плотность, г/м2
68	Модуль упругости при растяжении, ГПа
3000	Прочность при растяжении, МПа
210	Потребление смолы, г/м2
90	Угол нашивки, град
2	Шаг нашивки
50	Плотность нашивки

Рассчитать Сбросить

Расчет соотношения матрица-наполнитель

Введите параметры

	Плотность, кг/м3
	Модуль упругости, ГПа
	Количество отвердителя, м.%
	Содержание эпоксидных групп,%_2
	Температура вспышки, С_2
	Поверхностная плотность, г/м2
	Модуль упругости при растяжении, ГПа
	Прочность при растяжении, МПа
	Потребление смолы, г/м2
	Угол нашивки, град
	Шаг нашивки
	Плотность нашивки

Рассчитать Сбросить

Соотношение матрица-наполнитель: [[2.9434073]]

edu.bmstu.ru

+7 495 182-83-85

edu@bmstu.ru

Москва, Госпитальный переулок , д. 4-6, с.3

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана